**166. What are Exceptions.**

1. Exception are runtime errors.

2. There are three types of exceptions:

a. Syntax Errors: Errors in syntax (Mistakes done while writing the code).

b. Logical Errors:

i) Wrong/ Incorrect logic. Program don’t give expected results.

ii) Debugger is used to detect these errors.

c. Runtime Errors.

3. Syntax Errors and Logical Errors are faced by programmer.

4. Runtime Errors are exceptions.

5. Syntax Errors are detected by compiler.

6. Runtime errors are faced by the users unlike other errors which are faced and handled by the programmer.

7. Exceptions occur due to external factors like mishandling of software by the user.

Example:

1. Bad Input.
2. Resource unavailability.

8. In case of exception we can guide the user through messages.

9. The runtime errors are handled using the exception handling.

**167. How to handle Exceptions.**

1. In java infinity is undefined.

2. Aim for handling exception is that program should be terminated smoothly and normally.

3. In computer the infinity is not defined.

4. Its our choice what to write in try block. Followed by try block there should be catch block.

6. Following is the syntax of try and catch block.

try

{

//code

}catch(Parameters) {

//Exception code

}

7. There can be multiple catch blocks for one try block.

8. In case of multiple catch blocks, the uppermost catch must be sub class of class below otherwise the sub class will get shadowed by its base class.

for example:

catch(sub-class)

{}

catch(class)

{}

9. Nested try-catch block or multiple try-catch block usage is a choice of ours.

10. finally block is an optional block. It contains code which definitely gets executed in any circumstances.

11. finally block is used for clean up process.

**168. Try and Catch Block.**

1. Exception are runtime errors.

2. Finally contains code that must be executed.

**169. Multiple and Nested Try Catch.**

1. Multiple or Nested it depends on structure and choice of programmer.

**170. Class Exception.**

1. There are built in exception class in java with methods.

2. We can define our own exception class.

3. Object is the mother class of all the java classes.

4. hierarchy:

Object:

Throwable:

Error:

Exception:

(Checked Exception)

ClassNotFoundException.

IOException

FileNotFoundException

InterruptedException

NumberFormatException

(Unchecked Exception)

RuntimeException

ArithmeticException

IndexOutOfBoundException

NullPointerException

5. Exception is the parent class of all the exception classes.

6. All exceptions classes are categorized into two:

**Checked Exception**: You must handle them try and catch. These should be handled. Java force you to handle these exceptions.

**Unchecked Exception**: It is not mandatory to handle these exceptions.

7. First Comes the sub class exception’s catch bloc and then comes the parent class exception’s catch block. This is the hierarchy in multiple catch boxes.

8. Methods of Exception class:

a. String getMessage();

b. String toString();

c. void printStackTrace();

9. The feature of System.out.println(obj); is that whenever any object is given to it, it will call toString() method.

10. void printStackTrace() method used to get trace of method calls and how it leads to exception.

11. To write user defined exception you have to inherit Exception class into the user define exception class. If don’t inherit Exception class then it won’t be an Exception.

**171. Checked and Unchecked Exceptions.**

1. Compile r won’t force to handle the uncheck exception.

2. Learn the exception when you come across them.

3. To maintain log or report, printStackTrace() variation methods are used (Something like submitting report when app crashes.).

4. printStackTrace() is use to trace.

5. printStackTrace(PrintStream s): We can send the trace to some file.

6. toString: It gives exception name and exception.

7. getMessage: It gives exception.

8. It is not recommended to use printStackTrace().

9. We can define exception class by inheriting Exception class and override the methods of Exception.

**172. Throw vs Throws**

1. **throw:** This keyword is used to throw the exception.

2. Following is the way how the exception is propagated:

If exception is not handled then it will keeps passing the exception to the methods which calls method raising the exception. Until it reaches the main method and then JVM and the program crashes. We can get this trace chain using printStackTrace() method.

3. JVM calls main method and execute the program.

4. Throwing an exception is an alternative to returning in method. When exception is raised and you can’t return a value then you throw the exception.

5. Throw the object of exception.

6. Exception is a checked exception. You have to write the try and catch block for checked exception.

7. Try and catch block is written the method which is calling the method which is producing the exception.

8. If you are not handling the exception in the same line then we declare in the signature of method that this method thows exception.

method() throws Exception

{

throw new Exception(“Hello”);

}

throws is an adjective.

9. If a method doesn’t want to have an exception the throw it further.

10. It compulsory to write catch or finally block after try block.

11. Our own exception class must inherit the Exception class.

12. Depending our java project we can have our own exception classes and used them wherever we required.

**173. Demo: Throw and Throws.**

1. It is mandatory to use throws with throw keyword.

**174.Demo: Finally block.**

1. Finally block contains the code which will be executed with or without exception.

2. If a method is not handling the exception or throwing the exception to another method then it can use finally block to close the resources.

3. If you are not sure that you are handling all the exception then you must write the finally block.

**175. Try with Resources.**

1. Programs will be running inside the main memory and C.P.U will executing the program.

2. Program uses the memory in the form of stack and heap.

3. The program and its stack is one entity. It called as context of a program, its own area, its own boundary.

4. Heap is not a core part of a program its external to program.

5. Every object in java is created in heap.

6. All the things outside the program are resource for a program.

7. Method of using the resource is that when you need request them and when you don’t need you should return them otherwise it will be believed that the resource is in use.

8. Java has built-in garbage collector.

9. To make sure that cleanup is always done we use finally block.

10. Try with resources

11. You can create resource along with the try.

12 Syntax:

try(resources)

{

//code

return result;

}

13. In try and resource, we can create the resource along with the try and then the resource will be closed automatically once the method is terminates. Try takes its responsibility.

14. There is an built in interface in java.lang package as AutoClosable it will contains is of methods for automatically closing resources. This package gives us the idea about which resources can be closed automatically.

**176. Demo: Try with Resources.**

1. Resources in try block are separated with semicolon(;).

**177. Student Challenge: Stack Class**

1. Q1: Exception for stack.

a. if top == -1 and trying to delete from empty stack then it should print “Stack underflow”.

b. if top == 4 and trying to insert then it should print “Stack overflow.”

Define two classes, StackOverFlow Exception and Stack Under Flow Exception. Override toString() method.

Create class Stack having methods push and pop.